CNET Solution:

Xây dựng mạng với đỉnh phát là đỉnh S=1.Thêm một dỉnh thu giả T=n+1. Nối các đỉnh nhu cầu nối internet với đỉnh thu giả T với độ thông qua là 1. Các cung khác có độ thông qua là x. Có thể thấy một cách mắc cáp sẽ tương ứng với một luồng trên mạng này (mỗi dòng đơn vị là một sợi dây) và bài toán yêu cầu tìm luồng giá trị *k* với giá trị x là nhỏ nhất có thể.
Có thể chặt nhị phân theo giá trị x . Tuy nhiên ta có thuật toán nhanh hơn:
Khởi đầu x=1, mỗi lần tìm đường tăng luồng nếu không tìm được ta tăng độ thông qua các cạnh thêm 1 (mở rộng đường ống: x=x+1) cho đến khi tìm được đường tăng luồng. Đáp số sẽ là giá trị cuối cùng của x:

```
// Code bằng giải thuật Ford-Fulkerson
x:=1; // Khởi đầu độ thông qua
for(i=1;i≤k;++i) {
    while (!FindPath()) ++x;
    IncFlow();
}

// Code bằng giải thuật Dinitz:
mf:=0;
x:=1;
while (mf<k) {
    while (!BFS()) ++x;
    while (int delta=(++id,FindPath(S,oo)))
    mf+=delta;
}
```

Chú ý rằng các cung nối đến đỉnh T là các cung 1 chiều luôn có độ thông qua 1, các cung còn lại là các cung 2 chiều có độ thông qua x.
Thuật toán trên chạy nhanh hơn thuật toán tìm kiếm nhị phân thuần túy bởi vì trong thuật toán tìm kiếm nhị phân đơn giản thì mỗi khi có độ thông qua lớn nhất ta phải xây dựng lại luông từ đầu trong khi với thuật toán trên, ta chỉ tăng độ thông qua khi không tìm thấy đường và tất cả kết quả tăng luồng trước đó được giữ nguyên (không phải làm lại).

HLD Sample:

1) Heavy Light Decompose

```
// Input:
int n;
vector<int> adj[maxn];
// Output:
int pos[maxn];              // pos[u] = vị trí của u trong HLD DFS
int vertex[maxn];           // vertex[i]= tên đỉnh ở vị trí i
int par[maxn];              // par[u] = đỉnh cha của đỉnh u
int d_hld[maxn];            // d_hld[u]= Độ sâu của "đường nặng" chứa u
int head[maxn];             // Vị trí đầu "đường nặng" chứa u


void HLD(int u, int w) {
    pos[u] = ++id;
```

```
    vertex[id] = u;
    if (!w || 2 * s[u] < s[w])
        head[u] = id, d_hld[u] = d_hld[w] + 1;
    else
        head[u] = head[w], d_hld[u] = d_hld[w];
    for(int v : adj[u])
        if (v != w)
            HLD(v, u);
}

int s[maxn];        // s[u]=kích cỡ cây con chứa u
void DFS(int u, int dad) {
    s[u] = 1;
    int smax = 0, imax = -1;
    for(int i = 0; i < adj[u].size(); ++i) {
        int v = adj[u][i];
        if (v != dad) {
            par[v] = u;
            DFS(v, u);
            s[u] += s[v];
            if (smax < s[v])
                smax = s[v], imax = i;
        }
    }
    if (imax != -1)
        swap(adj[u][0], adj[u][imax]);
}
```

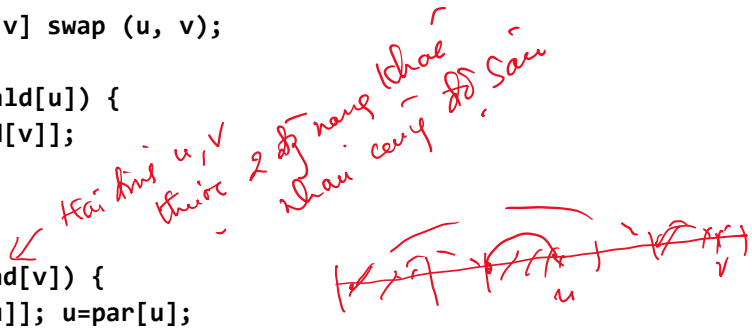
2) Tìm LCA:

```
int LCA(int u,int v) {
    if (d_hld[u]>d_hld[v] swap (u, v);

    while (d_hld[v]>d_hld[u]) {
        v = vertex[head[v]];
        v = par[v];
    }

    while (head[u]!=head[v]) {
        u=vertex[head[u]]; u=par[u];
        v=vertex[head[v]]; v=par[v];
    }

    if (pos[u]<pos[v]) return u;
    else return v;
}
```



3) Cập nhật tăng các cạnh trên đường đi từ u đến v

```
void IncEdges(int u,int v,int Delta) {
    if (d_hld[u]>d_hld[v]) swap(u, v);

    while (d_HLD[v]>d_HLD[u]) {
        update(1, 1, n, head[v], pos[v], Delta);
        v=par[vertex[head[v]];
    }
    while (head[u]!=head[v]) {
        update(1, 1, n, head[u], pos[u], Delta);
        u = par[vertext[head[u]];
```

```
        update(1, 1, n, head[v],head[v],Delta);
        v = par[vertext[head[v]];
    }
    if (pos[u]<pos[v]) {
        update(1,1,n,pos[u]+1,pos[v],Delta);
    } else {
        update(1,1,n,pos[v]+1,pos[u],Delta);
    }
}
```

ST quản lý cạnh
Nếu ST quản lý đỉnh thay thì pos[v]+1 bằng pos[u]

4) Truy vấn max các cạnh trên đường đi từ u đến v

```
int GetEdges(int u,int v) {
    int kq=-INF;
    if (d_hld[u]>d_hld[v]) swap(u, v);
    while (d_hld[v]>d_hld[u]) {
        kq=max(kq,get(1, 1, n, head[v], pos[v]);
        v=par[vertex[head[v]];
    }
    while (head[u]!=head[v]) {
        kq=max(kq,get(1, 1, n, head[u], pos[u]);
        u=par[vertex[head[u]];
        kq=max(kq,get(1, 1, n,head[v], head[v]);
        v=par[vertex[head[v]];
    }
    if (pos[u]<pos[v]) {
        kq=max(kq,get(1,1,n,pos[u]+1,pos[v]);
    } else {
        kq=max(kq,get(1,1,n,pos[v]+1,pos[u]);
    }
    return kq;
}
```

Chú ý
+ Có $\log_2 n$ cạnh nhẹ
=) Tối đa $\log_2 n$ nhảy về LCA.
+ Nếu lần nhảy duyệt ST
=) Chi phí $\log_2 n$.
=) Tổng TG 1 Truy vấn $O(\log^2 n)$

Câu Tím: Nếu gặp là 1 ST.